Marked-up version of the amended claims--additions are shown with double-underlines and deletions are shown with strike-throughs.

1. (Amended) A method for deleting entries from a directory in which directory information is stored in a set of database tables, comprising the steps of:

receiving a request to delete a directory entry;

responsive to receiving the ~~a~~ request to delete a directory entry, determining to tag ~~tagging~~ the directory entry for subsequent deletion by setting an attribute of the directory entry to a predetermined value;

updating ~~in~~ a first database table storing the attribute of the directory entry;

periodically searching for tagged directory entries in the first database table during a cleanup process interval; and

deleting references to the tagged directory entries throughout the set of database tables.

2. (Amended) The method as described in claim 1 wherein the directory entry is tagged by setting its creation time attribute to a given value.

3. The method as described in claim 2 wherein the given value is a null value.

4. (Amended) The method as described in claim 1, further including the steps of:

performing a search for directory entries that satisfy a search query; and

excluding tagged directory entries from search results that otherwise satisfy the search query.

5. (Amended) The method as described in claim 4 wherein the step of excluding tagged directory entries includes modifying an SQL query to exclude rows having a null change creation.

6. The method as described in claim 1 wherein the directory is a Lightweight Directory Access Protocol (LDAP) directory service and the database tables are managed by a relational database management service.

7. (Amended) The method as described in claim 1 wherein the first database table is an entry table.

8. The method as described in claim 7 wherein the set of database tables includes at least one attribute table storing information about an attribute.

9. (Amended) A method for deleting entries from a directory in which directory information is stored in a set of database tables, comprising the steps of:

receiving a request to delete a directory entry;

responsive to receiving the ~~a~~ request to delete a directory entry, determining to tag ~~tagging~~ the directory entry for subsequent deletion by setting an attribute of the directory entry to a predetermined value;

updating ~~in~~ a first database table storing the attribute of the directory entry;

responsive to a search for directory entries that satisfy a search query, excluding tagged directory entries from search results that otherwise satisfy the search query;

periodically searching for tagged during entries during a cleanup process interval; and

deleting references to the tagged directory entries throughout the set of database tables.

10. The method as described in claim 9 wherein the directory entry is tagged by setting its creation time to a given value.

11. The method as described in claim 10 wherein the given value is a null value.

12. (Amended) The method as described in claim 9 wherein the first database table is an entry table.

13. The method as described in claim 12 wherein the set of database tables includes at least one attribute table storing information about an attribute.

14. (Amended) A method for searching a database from a directory service, comprising the steps of:

receiving a search query;

responsive to a search for directory entries that satisfy the ~~a~~ search query, excluding given directory entries from search results that otherwise satisfy the search query, wherein a ~~the~~ given directory entry is a directory entry that has ~~entries identify database entries that have~~ been tagged for deletion by setting an attribute of the given directory entry to a predetermined value; and

returning the search results.

15. The method as described in claim 14 where in the directory service is a Lightweight Directory Access Protocol (LDAP) directory service and the database tables are managed by a relational database management service.

16. (Amended) A computer program product in a computer-readable medium for deleting entries from a directory in which directory information is stored in a set of database tables, comprising:

means for receiving a request to delete a directory entry;

means for determining, responsive to receiving the request to delete a directory entry, to tag the directory entry for subsequent deletion by setting an attribute of the directory entry to a predetermined value;

means for updating a first database table storing the attribute of the directory entry;

means for periodically searching for tagged directory entries in the first database table during a cleanup process interval; and

means for deleting references to the tagged directory entries throughout the set of database tables

~~means responsive to a request to delete a directory entry for tagging the directory entry in a first table;~~

~~means for periodically searching for tagged entries in the first table during a cleanup process interval, and~~

~~means for deleting references to the tagged entries throughout the set of database tables~~.

17. The computer program product as described in claim 16, further including:

means responsive to a search for directory entries that satisfy a search query for excluding tagged entries from search results that otherwise satisfy the search query.

18. The computer program product as described in claim 17 wherein the search query is a Lightweight Directory Access Protocol (LDAP) directory service query.

19. (Amended) A directory service, comprising:

a directory organized as a naming hierarchy having a plurality of entries each represented by a unique identifier;

a relational database management system having a backing store for storing directory data in a set of database entries; and

means for deleting entries from the directory, comprising:

means for determining, responsive to receiving the request to delete a directory entry, to tag the directory entry for subsequent deletion by setting an attribute of the directory entry to a predetermined value;

means for updating a first database table storing the attribute of the directory entry;

means for periodically searching for tagged directory entries in the first database table during a cleanup process interval; and

means for deleting references to the tagged directory entries throughout the set of database tables;

~~means responsive to a request to delete a directory entry for tagging the directory entry in a first table;~~

~~means for periodically searching for tagged entries in the first table during a cleanup process interval;~~

means responsive to a search for directory entries that satisfy a search query for excluding tagged entries from search results that otherwise satisfy the search query.

18. The computer program product as described in claim 17 wherein the search query is a Lightweight Directory Access Protocol (LDAP) directory service query.

19. (Amended) A directory service, comprising:

a directory organized as a naming hierarchy having a plurality of entries each represented by a unique identifier;

a relational database management system having a backing store for storing directory data in a set of database entries; and

means for deleting entries from the directory, comprising:

means for determining, responsive to receiving the request to delete a directory entry, to tag the directory entry for subsequent deletion by setting an attribute of the directory entry to a predetermined value;

means for updating a first database table storing the attribute of the directory entry;

means for periodically searching for tagged directory entries in the first database table during a cleanup process interval; and

means for deleting references to the tagged directory entries throughout the set of database tables;

~~means responsive to a request to delete a directory entry for tagging the directory entry in a first table;~~

~~means for periodically searching for tagged entries in the first table during a cleanup process interval;~~

~~means for deleting references to the tagged entries throughout the set of database tables;~~ and

means responsive to a search for directory entries that satisfy a search query for excluding tagged directory entries from search results that otherwise satisfy the search query.

20. The directory service as described in claim 19 wherein the directory is compliant with the Lightweight Directory Access Protocol (LDAP).

II. General Remarks Concerning This Response

Claims 1-20 are currently pending in the present application. Claims 1, 2, 4, 5, 7, 9, 12, 14, 16, and 19 have been amended in this response; no claims have been added or canceled. Reconsideration of the claims is respectfully requested.

The previous rejections under 35 U.S.C. § 103(a) have been withdrawn and new rejections have been substituted therefor.

III. Summary of Present Invention

The present invention is a method for deferred deletion of entries in a directory service backing store. Although shown as a preferred embodiment within the specification, the invention is not limited to a Lightweight Directory Access Protocol (LDAP) directory service provided with a DB/2 backing store. As stated in the specification, the principles of the present invention may be practiced in other types of directory services, e.g., X.500, and using other relational database management systems, e.g., Oracle, Sybase, Informix, etc., as the backing store.

In either the present invention or the prior art, an entry in an LDAP directory is deleted using an SQL statement. In the prior art, the directory server responds to the delete entry statement by instituting a global lock on the database tables to ensure that data in those tables cannot be modified while the entry is being deleted from the directory. In contrast, the present invention provides an enhanced delete operation whereby the entry is marked for deletion, and the actual deletion is completed at a later time.

More specifically, the present invention is a method for deleting an entry from a directory in which directory

information is stored in a set of database tables; the deletion is initiated in response to a request to delete a directory entry. In response, the directory entry is tagged in some manner as being a deleted entry, preferably by setting the entry's creation time to a null value. If a search query is received thereafter, the method excludes tagged entries from search results that would otherwise satisfy the search query. At a periodic interval, the routine then searches for tagged entries, and references to the tagged entries are then deleted throughout the set of database tables. In this manner, the completion of the entry deletion operation is deferred to enable directory queries to be processed even if deleted entries have not yet been fully expunged from the directory.

IV. 35 U.S.C. § 103(a)–Obviousness–*Kennedy*

The Office action has rejected claims 1-5, 7-14, 16, 17, and 19 under 35 U.S.C. § 103(a) as unpatentable over Kennedy, "System and method for managing electronic mail messages using a client-based database", U.S. Patent No. 6,134,582, filed 05/26/1998, issued 10/17/2000. This rejection is respectfully traversed.

The following is the rejection of independent claim 1 in its entirety:

> Re claims 1 and 16, Kennedy discloses a periodically search for tagged entries in the first table during a cleanup process interval, (col. 11, lines 48-col. 12, lines 8); and deleting references to the tagged entries throughout the set of database tables, (col. 3, lines 55-col. 4, lines 20).
>
> Kennedy does not explicitly disclose, "Responsive to a request to delete a directory entry, tagging the directory entry in a first table."

> However, Kennedy discloses message entries containing "delete" flags in the database (39) can be deleted from the server (49) after all message retrieval operations are completed. In particular, all message entries marked with a "delete" flag are located in response to walking the message entries in the database (39) and thereafter, are deleted from the server (40), (col. 11, lines 65-col. 12, lines 8).
>
> Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to responsive to a request to delete a directory entry [sic], tagging the directory entry in a first table, so the users can send and receive a large number of e-mail messages would like an effective way [sic] to process their e-mail without spending a lot of time sorting through their in-box, deleting, filing, forwarding, and responding to their messages.

Kennedy clearly does not disclose some of the claimed features of the present invention, notwithstanding the arguments presented by the rejection. Applicant first focuses on independent claim 1 (unamended):

> 1. A method for deleting entries from a directory in which directory information is stored in a set of database tables, comprising the steps of:
>
> responsive to a request to delete a directory entry, tagging the directory entry in a first table;
>
> periodically searching for tagged entries in the first table during a cleanup process interval; and
>
> deleting references to the tagged entries throughout the set of database tables.

As explained in the "Summary of the Invention" section of the Kennedy reference, the system that is disclosed in Kennedy provides a client-server method for managing electronic mail messages that are stored within a local, client-side database and a remote, server-side database. During a client-server session, message-related information is retrieved from the server and stored in the client-based database along with downloaded messages. Indications are provided in the client-side database as to whether a message has been

previously downloaded from the server and whether a copy of a message has been left on the server. Expiration times can be set so that messages are deleted over time. In essence, the system allows some of the information to be stored at the client and/or the server, and the system acts to synchronize the information in the local database and the remote database. Additional detail about the system of Kennedy is provided below within portions of the reference that have been recited by this response in arguments against the rejections.

The rejection admits that "Kennedy does not explicitly disclose" the first element of unamended claim 1, and the rejection then proceeds to discuss the delete flags in Kennedy. For example, against the first element of unamended claim 1, the rejection cites Kennedy at column 11, line 48, to column 12, line 8, in support of an argument that Kennedy discloses some portion of the first element, thereby drawing an analogy between tagging message entries with delete flags in Kennedy and the tagging operation in the present invention. However, the rejection fails to provide an explanation as to how Kennedy discloses an operation "responsive to a request to delete a directory entry" as required by the claim language. It is important to note that all of the claims in the present application are directed to a directory service or a method or a computer program product that involves a directory, but Kennedy does not mention a directory or a directory service even once. Kennedy does not disclose a directory or a directory service, so it is not possible for Kennedy to disclose the element "responsive to a request to delete a directory entry, tagging the directory entry in a first table" as recited in the first element of unamended independent claim 1.

To further the prosecution of the present patent application and not to amend around Kennedy, Applicant has amended the independent claims to clarify the operations that are performed with respect to directory entries. Amended independent claims 1, 9, 16, and 20 now include a specific recitation of elements to set an attribute of a directory entry to a predetermined value and updating a database table in which the attribute is stored. Amended independent claim 14 now includes a specific recitation of excluding given directory entries that have been tagged by setting an attribute of the given directory entry. Independent claims 9 and 14 continue to recite the feature of performing a search while excluding tagged directory entries.

Dependent claims 2-5, 7, and 8 are dependent upon independent claim 1 and include the elements of claim 1. Hence, because Kennedy does not disclose all of the features of independent claim 1 as required by a proper obviousness rejection, these references are also deficient against claims 2-5, 7, and 8.

Applicant also provides additional arguments concerning other deficiencies in the rejection of other claims besides independent claim 1. The following arguments essentially show that the portions of Kennedy that have been cited against specific claim elements do not disclose the claim elements as argued by the rejection.

With respect to dependent claim 2, the rejection cites two sections (column 11, line 48, to column 12, line 8; and column 13, line 33, to column 14, line 68) of Kennedy in support of an argument that Kennedy discloses claim 2, which reads: "wherein the directory entry is tagged by setting its creation time to a given value". Applicant notes that, in almost two columns of patent text, there are no features of

the disclosed system that are even slightly similar to the claimed feature. The cited portions of Kennedy mention that a date and time field is compared against a preset period of time, and based on the results of the comparison, a "delete" flag might be set. However, as should be apparent by a simple reading of the passage in Kennedy, the delete flag is separate from the date and time field; the date and time field is not used to indicate that the entry should be deleted.

With respect to dependent claim 3, the rejection cites the same two sections of Kennedy that were applied to dependent claim 2 (column 11, line 48, to column 12, line 8; and column 13, line 33, to column 14, line 68) in support of an argument that Kennedy discloses the claim, which reads: "wherein the given value is a null value". Claim 3 is dependent on claim 2; Kennedy does not disclose the use of a creation time as a delete tag, as required by claim 2, and Kennedy does not disclose that the creation time is set to a null value, as required by claim 3.

Against the first element of dependent claim 4, which comprises the element of "performing a search for directory entries that satisfy a search query", the rejection again cites a portion of Kennedy at column 13, line 66, to column 14, line 47; as noted above, Kennedy does not disclose the use of directory entries. Against the second element of dependent claim 4, which comprises the element of "excluding tagged entries from search results that otherwise satisfy the search query", the rejection cites a portion of Kennedy at column 21, line 54, to column 23, line 18. The basis of argument for this rejection is entirely unfounded because the rejection cites approximately one and a half columns of patent text that discusses a method of assembling a message that comprises multiple message parts; nothing in the cited portion even

remotely approaches a feature that is similar to the claimed element of "excluding tagged entries ...". Applicant also notes that the cited portion appears to be randomly selected and randomly applied against the claim element because the cited portion includes the concluding paragraphs of the patent and the first several lines of the first claim.

Against dependent claim 5, which states that "the step of excluding tagged entries includes modifying an SQL query to exclude rows having a null change creation", the rejection cites a portion of Kennedy at column 8, line 51, to column 9, line 63, which is provided immediately below (Applicant herein recites the entire section of more than a column of patent text to emphasis what appears to be a random selection of a portion of the reference because it has no relevance to the claimed feature):

> In FIG. 3, a remote computer 49 operates as a server and generally includes an e-mail server application 110, a local store 115, and a client manager control 120. In an exemplary embodiment, the server 49 is a POP3 mail server, but it will be appreciated that the present invention is not limited to this type server. In the exemplary embodiment, the client 20 includes a local message store 38, a database 39, an e-mail program module 36, and a message manager program module 37 for facilitating message management and operation of the database 39.
>
> With respect to the exemplary embodiment, the client 20 provides two modes of operation in connection with the server 49. These modes are a default mode and a "leave on server" mode. In the default mode, the client 20 sends a delete command to the server 49 to delete a message from the server 49 after the message has been downloaded to the client 20. In the "leave on server" mode, the client does not send a delete command to the server 49 after the message has been downloaded to the client 20, thereby allowing the message to remain on the server 49 although the message has been downloaded. The mode of operation generally is selected based on user-preference. Advantageously, the present invention optimizes the management of messages when the client 20

is in the "leave on server" mode, as will be described below in connection with FIGS. 4-8.

The server 49 houses any e-mail messages from clients in the local store 115 while awaiting transmission to an appropriate destination. The e-mail server application 110 forwards messages over the WAN 52 from a sender client (not shown) to the client 20, upon request by the client 20. The client manager control 120 is a program used to set up computer systems, such as clients 1, 11a, 11b, 11c (FIG. 1), and 20 (FIG. 2) on the network. The client manager control 120 can also specify the addresses of the computer systems located on the network. In addition, the client manager control 120 typically facilitates the management of incoming and outgoing messages on the server. When a request for a message is made by the client 20 to the server 49, the e-mail server application 110 on the server 49 responds by retrieving the message from the local store 115 on the server 49 and by transmitting the message over the WAN 52 to the client 20. The message is then downloaded into the local message store 38 located at the client 20. The local message store 38 houses all downloaded messages from the server 49.

During the download operation, data fields are populated within the database 39 with message-related information associated with the downloaded message. The information includes a unique identifier for identifying the message, a session identifier for indicating the particular order in which the message is retrieved from the server, a message size and other message-related information that will be described in greater detail herein below with respect to FIGS. 4-8. The e-mail program module 36 provides facilities for creating, addressing, sending, receiving, and forwarding messages, while the message manager program module 37 manages messages during download and deletion operations utilizing the database 39. The use of the database 39 is described in greater detail in connection with FIGS. 4a-4k, collectively described as FIG. 4.

With continuing reference to FIGS. 1-3 and now turning to FIGS. 4a-4k, a client-based database used in connection with the exemplary program module 37 is illustrated. FIGS. 4a-4k illustrate a client-based database for archiving messages in accordance with an exemplary embodiment of the present invention.

The database 39 can include multiple data fields, organized within an array structure, for maintaining message-related information. To support download and

> delete operations, typical data fields of the database include: a session identifier (session ID) 200, a unique identifier (UIDL) 205, a message size 210, an entry identifier (EID) 215, a receive date and time 220, which is the local machine time, an "on server" flag 225, a "download" flag 230, and a "delete" flag 235. Message re-assembly operations can be supported by adding to the database structure certain data fields corresponding to portions of a MIME-compatible message, such as a message group identifier (message group ID) 240, a message part number 245, and a total parts number 250.

It is entirely unclear how this passage discloses anything remotely related to the modification of an SQL query.

With respect to independent claims 9, 14, and 19, these claims are similar to independent claim 1 and dependent claim 4 in that claims 9 and 19 are a combination of the elements of claims 1 and 4 and claim 14 is a subset of the combination of claims 1 and 4. The rejection merely cites the same portions of Kennedy against claims 9, 14, and 19 that were cited against claims 1 and 4. Applicant has refuted these arguments above with respect to claims 1 and 4, and Applicant maintains that independent claims 9, 14, and 19 are not obvious in view of Kennedy for the same reasons as claims 1 and 4.

With respect to independent claim 16, this claim is similar to independent claim 1 in that independent claim 1 is a method claim, whereas independent claim 16 is directed to a computer program product comprising means for performing the steps of a process similar to that claimed in claim 1. Independent claim 16 has been rejected using the same argument as independent claim 1, and Applicant maintains that independent claim 16 is not obvious in view of Kennedy for the same reasons that were provided above for independent claim 1.

Dependent claims 10-13 and 17 were rejected using the same arguments as dependent claims 2-4, 7, and 8. Applicant maintains that dependent claims 10-13 and 17 are not obvious

in view of Kennedy for the same reasons that were provided above for dependent claims 2-4, 7, and 8.

Examiner bears the burden of establishing a *prima facie* case of obviousness.

The examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). Only when a *prima facie* case of obviousness is established does the burden shift to the applicant to produce evidence of nonobviousness. *In re Oetiker*, 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Rijckaert*, 9 F.3d 1531, 1532, 28 U.S.P.Q.2d 1955, 1956 (Fed. Cir. 1993). If the Patent Office does not produce a *prima facie* case of unpatentability, then without more the applicant is entitled to the grant of a patent. *In re Oetiker*, 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Grabiak*, 769 F.2d 729, 733, 226 U.S.P.Q. 870, 873 (Fed. Cir. 1985). In response to an assertion of obviousness by the Patent Office, the applicant may attack the Patent Office's *prima facie* determination as improperly made out, present objective evidence tending to support a conclusion of nonobviousness, or both. *In re Fritch*, 972 F.2d 1260, 1265, 23 U.S.P.Q.2d 1780, 1783 (Fed. Cir. 1992).

With respect to claims 1-5, 7-14, 16, 17, and 19, Kennedy does not disclose the claimed invention nor does Kennedy provide any suggestion to motivate one having ordinary skill in the art to modify the system of Kennedy to reach the claimed invention. In fact, the rejection appears to disregard entire claim elements without justification. In general, the rejection does not point out the necessary teachings,

suggestions, or incentives to reach the claimed invention. Hence, the rejection of claims 1-5, 7-14, 16, 17, and 19 does not establish a *prima facie* case of obviousness based on the prior art. Therefore, the rejection of claims 1-5, 7-14, 16, 17, and 19 under 35 U.S.C. § 103(a) has been shown to be insupportable, and these claims are patentable over the applied prior art. Applicant requests the withdrawal of the rejection.

V. 35 U.S.C. § 103(a) – Obviousness – *Kennedy* in view of *Byrne et al.*

The Office action has rejected claims 6, 15, 18, and 20 under 35 U.S.C. § 103(a) as unpatentable over Kennedy, "System and method for managing electronic mail messages using a client-based database", U.S. Patent No. 6,134,582, filed 05/26/1998, issued 10/17/2000, in view of Byrne et al., "Lightweight Directory Access Protocol (LDAP) Directory Server Cache Mechanism and Method", U.S. Patent No. 6,347,312 B1, filed 11/05/1998, issued 02/12/2002. This rejection is respectfully traversed.

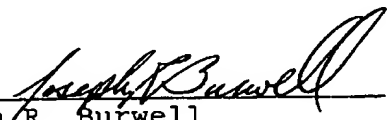
The rejection relies on Byrne et al. for disclosing an LDAP directory and a relational database management system. Byrne et al. clearly discloses an LDAP directory service having a relational database management system as a backing store. However, Byrne et al. does not provide any disclosure that remedies the deficiencies of Kennedy with respect to the independent claims on which claims 6, 15, 18, and 20 rely. Hence, claims 6, 15, 18, and 20 are also allowable for the same reasons provided above for independent claims 1, 14, 16, and 19.

VI. Conclusion

It is respectfully urged that the present patent application is patentable, and Applicant kindly requests a Notice of Allowance.

For any other outstanding matters or issues, the examiner is urged to call or fax the below-listed telephone numbers to expedite the prosecution and examination of this application.

DATE: March 21, 2003 Respectfully submitted,



Joseph R. Burwell
Reg. No. 44,468
ATTORNEY FOR APPLICANT

Law Office of Joseph R. Burwell
P.O. Box 28022
Austin, Texas 78755-8022
Voice: 866-728-3688 (866-PATENT8)
Fax: 866-728-3680 (866-PATENT0)
Email: joe@burwell.biz